## AMENDMENTS TO THE SPECIFICATION

Please amend the specification as follows:

Amend the paragraph on page 2, lines 4-6, as follows:

This application claims the benefit of and incorporates by reference U.S. Provisional Application No. 60/260,494, filed January 8, 2001, titled "Code Generation By a Matching and Satisfiability Search ~~System and Method for Program Verification~~".

Amend the paragraph on page 2, lines 9-10, as follows:

This application is related to and incorporates by reference U.S. ~~Patent (Application)~~ Serial No. 09/176,950 filed October 22, 1998, ~~by J. Saxe et al.~~ now U.S. Patent No. 6,343,376.

Amend the paragraph on page 11, line 24, through page 12, line 6, as follows:

The invention contemplates that a set of goal terms specifies what the corresponding near-optimal code sequence is supposed to do (they are the *exprlist* or R-values). The set of goal terms is the set of right-hand-side expressions of the guarded multi-assignment that is presented to the code sequence generator, together with the guard expression *G*. A programmer may enter, for example, a multi-assignment with goal terms at the right-hand-side as follows: *reg6, reg7:=reg6*(reg7+8),6*. Values of the set of goal terms (the values that computation of the goal terms is supposed to produce) are computed into targets listed by designators (the aforementioned *targetlist*, also called L-values). Thus, the left-hand-side of the multi-assignment indicates the list of designators which are the (register or memory) destinations of the result values. The output of the code sequence generator would be the near-optimal code sequence (program instructions or machine code) for computing the values of the goal terms and storing them in the specified destination registers. Figs. 1a and 1b illustrate the automatic code generator and inner loop of the generation process (referenced as 10a and 10b).

Amend the paragraph on page 13, lines 18-28, as follows:

The second phase is the constraint generator 30 and solver 32, also known herein as the planning phase with satisfiability search. The other input to the second phase is a description of the target architecture 23 (including, latencies of various operations, multiple issue restrictions,

etc.). The constraint generator 30 formulates this remaining question as a boolean satisfiability problem. Namely, for a fixed cycle budget, the constraint generator formulates as a boolean satisfiability problem the question of whether the expressions in the original GMA can be computed within the cycle ~~budge~~ budget using the target architecture. This problem is then given to a satisfiability solver 30. The boolean satisfiability solver (satisfiability search) is used to find a solution (the ways) or establish that no solution exists. The steps of the second phase, i.e., the constraint generation and satisfiability search steps, are repeated for various cycle budgets until an optimal machine program is found (i.e., until the minimum cycle solution is determined).